LayerSlayer

Transient

LayerSlayerBlanket User's Manual

Matthew R. Begley
Mechanical Engineering, University of California, Santa Barbara

(Dated: 24 December 2012)

This document provides user instructions for the multilayer analysis code ***LayerSlayerBlanket***. The framework is general for an arbitrary number of layers, which are assumed to be elastic, isotropic layers described by a modulus, coefficient of thermal expansion, thermal conductivity, a reference (strain-free) temperature, growth strain, and the temperatures at the top and bottom of each layer (which can be solved for using steady-state thermal analysis). The code predicts the following:

- steady-state temperature distribution through the layers (for either two prescribed temperatures, or a prescribed flux with one temperature)
- deformation of the multilayer, assuming plane strain or biaxial deformation
- stresses at the top and bottom of each layer (steady-state solution is a linear stress distribution through each stack)
- energy release rate for debonding, assuming biaxial curvature ahead of and behind the crack
- energy release rate for debonding, assuming biaxial curvature ahead of the crack and zero stress in the direction of the crack (but non-zero stress in the out-of-plane direction, dictated by the curvature ahead of the crack).
- energy release rate for debonding under applied moment, assuming plane-strain deformation ($\varepsilon_z = 0$) everywhere.

I. GETTING STARTED

LayerSlayer Blanket is written in *Mathematica*. The source code (which defines commands used in the analysis) is `LSBencrypt`, which is read ("loaded") into any *Mathematica* notebook. The source code is loaded using the *Mathematica* commands:

```
SetDirectory["/Users/Begley/Desktop/LSB FINAL"]
<< LSBencrypt‘,
```

where the pathway in the `SetDirectory` command should be set to the name of the folder in which the file `LSBencrypt` is stored. **NOTE: reading in the encrypted files will generate error messages pertaining to function definitions: *these can be ignored*.** The file `LSB Example` is a conventional *Mathematica* notebook which illustrates how various analyses are conducted.

II. OVERVIEW

The analysis is conducted for a stack of blanket films, as shown in Figure 1. The films are assumed to be infinite in the (x,z) plane, with a semi-infinite crack that grows along an interface that lies on a specific (x,z) plane, with a crack tip that is aligned

with the $z-$direction; energy release rates are computed for the steady-state growth scenario, i.e. when the crack is long enough so as to not influence the results. The problem is essentially one-dimensional, such that behavior in the (x,y) plane is analyzed: in the third dimension (out-of-plane to the analysis), one can either assume plane strain deformation (i.e. $\varepsilon_z = 0$) or biaxial deformation (i.e. $\varepsilon_z = \varepsilon_x$).

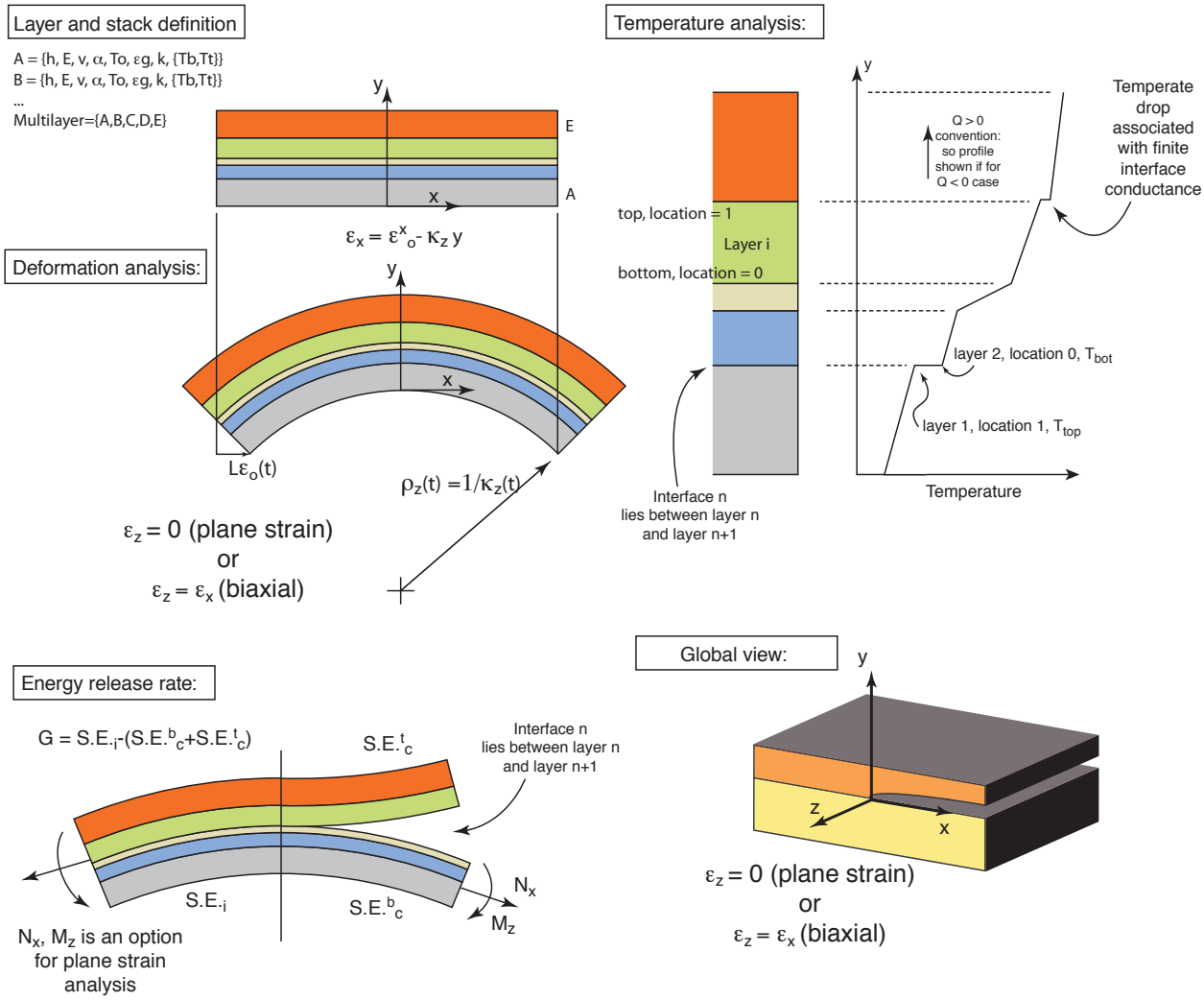

FIG. 1. Schematic illustrations of the geometry, variables and assumptions that underlie *LayerSlayerBlanket*

Temperatures are assumed to be uniform in the plane of the blanket, i.e. in the (x,z) plane, with gradients occurring only in the through-thickness direction, i.e. in the $y-$direction. The thermal analysis solves for the steady-state distribution through the stack, which yields a piece-wise linear temperature profile: if finite conductance is specified at the interfaces, the temperature is not continuous across the interface but rather experiences jumps proportional to the inverse of the interface conductance: hence, the temperatures on either side of the interface (i.e. bottom of one layer and top of layer beneath it) are retained as possibly independent variables. Temperatures on either side of an interface will be identical if the interface conductance is sufficiently high.

When analyzing cracking, it is assumed that the crack has no influence on the temperature distribution, i.e. the intact stack is analyzed. While cracking would naturally alter the heat flow in the cracked region, it is assumed that cracking events are sufficiently fast so as to not allow for changes associated with debonding. That is, the scenario being analysis consists of a steady-state crack growing at speeds faster than thermal transport. (The analysis of initiation of crack growth, involving a different temperature profile behind the crack due to debonding, requires a 2D thermal analysis to account for spatial gradients

in the $x-$direction.)

The key assumption is that the strain distribution in bonded layers is given by:

$$\varepsilon_x(y) = \varepsilon_o^x - \kappa_z y + \alpha(T(y) - T_o(y)), \tag{1}$$

where ε_o^x is the elongation in the $x-direction$ at the bottom of the stack, κ_z is the curvature of the stack about the $z-axis$, α is the coefficient of thermal expansion in the layer, $T_o(y)$ is a reference temperature that has a linear distribution in a given layer, and $T(y)$ is the linear temperature distribution dictated by the temperature at the top and bottom of a given layer. This strain distribution is used to compute the resultant moment and normal force on any stack (including sub-stacks created by cracking), which are equated to applied moments/forces. This fundamentally determines the elongation and curvature of the stack; the stresses and strain energy densities are then computed using those deformation variables and the above strain definition.

A variety of scenarios can be addressed with regards to the behavior parallel to the crack front, including plane strain deformation ($\varepsilon_z = 0$), or biaxial deformation ($\varepsilon_z = \varepsilon_x$). In the 'classic' ERR computation, it is assumed that cracking alleviates curvature about the $z-$axis, whereas the curvature about the $x-$axis is fixed to be the same ahead of the crack.

III. LAYER AND STACK DEFINITION

A. Individual layer definition

Each is layer given a name and the list of its properties, defined according to:

$$\texttt{Layer = } \{h, E, \nu, \alpha, \{T_o^{bot}, T_o^{top}\}, \varepsilon_g, k, \{T_b, T_t\}\}, \tag{2}$$

where `Layer` is the name of the layer (can be anything!), h is the layer thickness, E is the layer modulus, ν is the layer's Poisson ratio, $\{T_o^{bot}, T_o^{top}\}$ are the bottom and top reference temperatures for that layer (i.e. there is a linear distribution of reference temperature, which defines the zero thermal strain temperature for that location), ε_g is the growth strain in the layer, k is the thermal conductance of the layer, and $\{T_b, T_t\}$ are the current temperature of the layer at the top and bottom of the layer.

The order must be as prescribed above. The temperature distribution in each layer is linear, as would result from a steady-state thermal analysis. **If a thermal analysis step is conducted, the layer definitions are overwritten such that $\{T_b, T_t\}$ correspond to the results of the steady-state analysis: the other properties are not changed.** If desired, one can re-define the reference temperatures to be those associated with the steady-state temperatures found via the thermal analysis, and re-set the current temperature to be the temperature of interest. The default is that the reference temperature remain unchanged after thermal analysis, while the current temperatures are set to those solved for in the thermal analysis.

B. Stack/multilayer definition

The multilayer is defined as a list of layers, as in:

$$\texttt{Multilayer = } \{Substrate, BondCoat, Coating\}, \tag{3}$$

where *Substrate*, *BondCoat* and *Coating* are previously defined layers. ***The order must be from bottom to top***, i.e. the order in the *Multilayer* list indicates the relative position of the layers. Note that you can define the multilayer with any name (involving layers with any name), e.g. `CaseA = {Layer1, Layer2}`.

For the thermal analysis, the analysis incorporates interface elements *with zero thickness* to allow for temperature drops between layers associated with finite values of interface conductance. For the thermal analysis commands, one must provide a list of interface conductances, with the numbering/ordering being from bottom to top: e.g.:=

$$\texttt{Interfacesconductance = } \{k_i^{12}, k_i^{23}\}, \tag{4}$$

where k_i^{12} is the interface conductance between layers 1 and 2, k_i^{23} is the interface conductance between layers 2 and 3, and so forth. Again, note that this list of interface properties can be given any name, as in `IfaceEx` = $\{10^6, 10^3\}$, etc. Setting the interface conductances to a large number (in comparison to the effect layer conductances, characterized by k/h) results in nearly

perfect conduction, with (piece-wise) continuous temperatures throughout the stack.

For cracking analyses, one must provide (to the ERR commands, see below) the interface of interest (for which the energy release rate is to be computed): these are numbered, with interface n falling between layers n and $n+1$. Thus, the limits on possible interface numbers run from 1 to $N-1$, where N is the number of layers. For the purely plane strain ERR, you can also specify applied moments and loads.

IV. THERMAL BOUNDARY CONDITIONS: STEADY-STATE ANALYSIS

Two types of steady-state analyses are included, distinguished by the thermal boundary conditions that are imposed. **NOTE: running the thermal analysis command overwrites the layer definition, replacing the previous temperatures at the top and bottom of each layer with those resulting from the thermal analysis. Hence, there is no need to redefine the layer definitions to account for the steady-state temperature distribution, as the new definitions have the proper values inserted. If you would like to calculate the deformation/ERR for a temperature distribution that is not steady-state, simply insert the proper temperatures at the top and bottom but do not run the thermal analysis.**

A. Prescribing two temperatures

The function `TempSol[]` requires two temperatures to be specified within the entire multilayer stack. One can specify temperatures either at the bottom or the top of any given layer. The boundary conditions are specified as:

$$\texttt{TempBCs = } \{\{Layer\ \#,\ location,\ T_1\},\{Layer\ \#,\ location,\ T_2\}\}, \tag{5}$$

where $location = 0$ for temperature specified at the bottom of the layer, and $location = 1$ for temperature specified at the top of the layer. Again, note that you can call this list according to any name you choose. (I.e., the names used for lists here are just examples.) When interface conductance is essentially infinite, it doesn't matter if you specify the temperature at the top of one layer (i.e. the bottom of the interface) or the temperature at the bottom of the adjacent layer (i.e. at the top of the interface). That is, for infinite interface conductance, specifying either $\{1,1,T_1\}$ or $\{2,0,T_1\}$ yields the same result. When interface conductance is finite, such that there is a temperature drop across the interface, you must specify the temperature at the proper side of the interface.

B. Prescribing the heat flux and one temperature

The function `TempSolQ[]` requires one temperature and the heat flux to be specified. One can specify the temperature either at the bottom or the top of a given layer. The boundary conditions are specified as:

$$\texttt{TempBCs = } \{\{Layer\ \#,\ location,\ T_1\},Q\}, \tag{6}$$

where $location = 0$ for temperature specified at the bottom of the layer, and $location = 1$ for temperature specified at the top of the layer. Q is the heat flux in the layer, defined as positive from bottom to top. (If the top temperature is higher than the bottom temperature, then $Q < 0$.) Again, when interface conductance is essentially infinite, it doesn't matter if you specify the temperature at the top of one layer (i.e. the bottom of the interface) or the temperature at the bottom of the adjacent layer (i.e. at the top of the interface). That is, for infinite interface conductance, specifying either $\{1,1,T_1\}$ or $\{2,0,T_1\}$ yields the same result. When interface conductance is finite, such that there is a temperature drop across the interface, you must specify the temperature at the proper side of the interface.

V. SUBROUTINES/FUNCTIONS

A. Temperature Analysis

There are two function calls for steady-state thermal analysis, depending on whether one prescribes two temperatures, or one temperature with the heat flux. These commands are:

$$\texttt{TempSol[}Multilayer, InterfaceProps, BCs\texttt{]} \tag{7}$$

$$\texttt{TempSolQ[}Multilayer, InterfaceProps, BCs\texttt{]} \tag{8}$$

where *Multilayer*, *InterfaceProps* and *BCs* are the names of the lists defined earlier. (E.g., according to the above example, *BCs* is replaced with `TempBCs`.) These commands return a list of position/temperature pairs, corresponding from bottom to top of the stack. That is, they return the following:

$$\{\{0,T^1_{bot}\},\{h_1,T^1_{top}\},\{h_1,T^2_{bot}\},\{h_1+h_2,T^2_{top}\},...\} \tag{9}$$

That is, a list of absolute global positions in the stack and temperatures at those positions is returned; positions are repeated at interfaces, such that there are discrete points for the top and bottom of the interface. The command `ListPlot` creates a temperature-position plot, illustrating any temperature jumps at the interfaces that are a result of finite interface conductance. Again, when the interface conductance is large, the temperatures are continuous across the interfaces.

NOTE: running the thermal analysis command overwrites the layer definition, replacing the previous temperatures at the top and bottom of each layer with those resulting from the thermal analysis. Hence, there is no need to redefine the layer definitions to account for the steady-state temperature distribution, as the new definitions have the proper values inserted. If you would like to calculate the deformation/ERR for a temperature distribution that is not steady-state, simply insert the proper temperatures at the top and bottom but do not run the thermal analysis.

B. Deformation

There are two subroutines to calculate multilayer deformation: one for biaxial deformation with no loading and one for plane deformation with an applied moment and axial force. In this calculation, one can specify a uniform strain parallel to the crack front, i.e. $\varepsilon_z = constant$. The biaxial deformation is found with:

$$\texttt{BiaxialDeformation[}Multilayer\texttt{]} \tag{10}$$

where *Multilayer* is the name of the stack that was defined previously by creating a list of layers. This function returns a pair (defined as a Mathematica list) that contains the elongation of the bottom of the stack, ε_o, and the curvature of the stack, κ, as in $\{\varepsilon_o,\kappa\}$. For the biaxial deformation case, these deformation quantities are calculated by setting the net resultant moment and the net resultant normal force equal to zero.

One can substitute an explicit list of the layers desired for analysis, as in:

$$\texttt{BiaxialDeformation[\{}Substrate,Coating\texttt{\}]}, \tag{11}$$

That is, you can define the multilayer in the function call itself, and avoid having to name/define additional multilayers if you want to chop out a layer.

Similarly, there is a separate function call for generalized plane strain deformation, in which one prescribes the applied moment, the applied axial resultant, and the value of the out of plane strain. This type of deformation is found with:

$$\texttt{PlaneDeformation[}Multilayer,M_a,N_a,\varepsilon_z\texttt{]} \tag{12}$$

where *Multilayer* is the name of the stack currently being analyzed, M_a is the value of the applied moment, N_a is the value of the applied axial force, and ε_z is the constant value of the strain in the $z-$direction.

C. Stresses

Stresses are found by providing the elongation and curvature of the stack in the $x-$ and $z-$directions. That is, one provides $\{\varepsilon_o,\kappa\}_x$ and $\{\varepsilon_o,\kappa\}_z$, where the direct total strain in these directions is given by:

$$\varepsilon_i = \varepsilon_o - \kappa y, \tag{13}$$

using the appropriate values for the elongation and curvature for the given direction. The command that computes stress is:

$$\texttt{GetStress[\{}\varepsilon_o,\kappa\texttt{\}}_x,\texttt{\{}\varepsilon_o,\kappa\texttt{\}}_z,Layer,dir,scale\texttt{]} \tag{14}$$

where $\{\varepsilon_o,\kappa\}_x$ define the total strain in the $x-direction$, while $\{\varepsilon_o,\kappa\}_z$ define the total strain in the $z-direction$, *Layer* is the property list, $dir = 1$ for σ_x (and $dir = 3$ for σ_z): the variable *scale* is simply a scale-factor that allows you to recover appropriate

units. E.g., $scale = 10^6$ yields stresses in MPa if all the units quantities are in base SI units. The command returns an ordered list, with each entry being a list of the position and stress at the bottom and top of the layers, as in:

$$\{....,\{\{y^i_{bot},S_{bot}\},\{y^i_{top},S_{top}\}\},...\} \tag{15}$$

where the positions (e.g. y^i_{bot}) are measured from the bottom of the stack. Thus, using `ListPlot` with the output of `GetStress` produces a plot of the stresses in the layers, with a different (linear) curve representing the stress in each layer. (See examples.)

D. Energy Release Rates

There are three commands to calculate the energy release rate for debonding:

$$\texttt{ClassicERR[}Multilayer\texttt{, \#]} \tag{16}$$

$$\texttt{BiaxialERR[}Multilayer\texttt{, \#]} \tag{17}$$

$$\texttt{PlaneStrainERR[}Multilayer\texttt{, \#, }M_a\texttt{]} \tag{18}$$

In all commands, *Multilayer* is the name of the list that has the properties of all layers in the stack, while the entry # denotes the interface number where debonding occurs. The interfaces are numbered from the bottom of the stack, such that interface i lies between the i and $i+1$ layers.

In the *ClassicERR* command, the deformation ahead of the crack is assumed to be biaxial curvature. Behind the crack, the layers are assumed to experience plane deformation, with the strain in the $z-$direction set to the value determined from the fully intact biaxial curvature. That is, it is assumed that debonding relieves the net resultants acting on the two sections behind the crack front, but does not relieve the stresses in the $z-$direction. This recovers the usual semi-infinite substrate result for a blanket film.

In the *BiaxialERR* command, the deformation is assumed to be biaxial both before and after debonding. It is assumed that the intact multilayer experiences biaxial curvature, and that the multilayers behind the crack also experience biaxial curvature.

In the *PlaneStrainERR* command, zero strain is imposed in the out-of-plane direction, both before and after debonding. The quantity M_a refers to the applied moment applied to both ends of the multilayer: this moment is applied to the whole stack ahead of the crack, and only to the bottom stack (i.e. the bottom stack formed by the debond) behind the crack front. Plane strain conditions are enforced (i.e. $\varepsilon_z = 0$) for all sections.

VI. ILLUSTRATION

A. Layer and stack definition

Consider a three layer system, with the following layer properties:

Layer	Thickness	Modulus	Poisson's Ratio	CTE	Ref. Temp	Growth strain	Thermal conductance	Bot., Top Temp
`Layer1`	0.003	$200x10^9$	0.3	$15x10^{-6}$	$\{70,70\}$	0	25	$\{0,0\}$
`Layer2`	0.002	$40x10^9$	0.2	$11x10^{-6}$	$\{70,70\}$	0	1	$\{0,0\}$
`Layer3`	0.001	$100x10^9$	0.2	$11x10^{-6}$	$\{70,70\}$	0	2	$\{0,0\}$

The temperatures at the top and bottom of the layer are set to zero since they will be calculated by the thermal analysis. In the above, the reference temperatures are uniform and equal to room temperature. The multilayer is defined as follows:

$$\texttt{Layer1 = }\{0.003,200.x10^9,0.3,15x10^{-6},\{70,70\},0,25,\{0.,0.\}\}; \tag{19}$$

$$\texttt{Layer2 = }\{0.002,40.x10^9,0.2,11x10^{-6},\{70,70\},0,1,\{0.,0.\}\}; \tag{20}$$

$$\texttt{Layer3 = }\{0.001,100.x10^9,0.2,11x10^{-6},\{70,70\},0,2,\{0.,0.\}\}; \tag{21}$$

$$\texttt{Multilayer =}\{\texttt{Layer1,Layer2,Layer3}\} \tag{22}$$

Thus, the layers are named appropriately from bottom to top, i.e. Layer1 is on the bottom, and Layer3 is on the top.

B. Thermal boundary conditions

First, consider the case where the temperature at the bottom of the stack is fixed to be 800^o, while the temperature at the top of the stack is fixed to be 1300^o. Let's assume that the interface between Layer1 and Layer2 has finite conductance, such that there is a temperature drop across that interface: the interface between Layer2 and Layer3 has a large conductance (nearly perfectly conductive), such that there will be no temperature drop. This is stated as:

$$\texttt{TwoTbcs = } \{\{1,0,800\},\{3,1,1300.\}\}\texttt{; (*layer number, 0-bot,1-top, temp*)} \tag{23}$$

$$\texttt{ifc = } \{10^4, 10^7\} \tag{24}$$

Second, consider the case where the top surface temperature is known to be 1300^o and the heat flux (from top to bottom) with is known to be $Q = 169244$, then we would specify:

$$\texttt{TQbcs = } \{\{3,1,1300\},-169244\}\texttt{; (*layer number, 0-bot,1-top, temp*)} \tag{25}$$

$$\texttt{ifc = } \{10^4, 10^7\} \tag{26}$$

where a negative heat flux is prescribed between the top temperature is known to be higher than the bottom temperature (i.e. heat flows from top to bottom, which is negative by convention).

C. Results from Temperature Analysis

The "two temperature boundary condition" thermal analysis is conducted by executing:

$$\texttt{Tout = TempSol[Multilayer, ifc, TwoTbcs]} \tag{27}$$

which returns the following:

$$\texttt{Heat flux: } \texttt{-169294.} \tag{28}$$

$$\{\{0.,800.\},\{0.003,820.315\},\{0.003,876.747\},\{0.005,1215.34\},\{0.005,1215.35\},\{0.006,1300.\}\} \tag{29}$$

Since the heat flux is defined as positive from bottom to top, and the top temperature is highest, the heat flows in this case from top to bottom, which is a negative heat flux according to the adopted convention. Note the $\approx 56^o$ temperature drop between Layer1 and Layer2. Here is a `ListPlot` of the result:

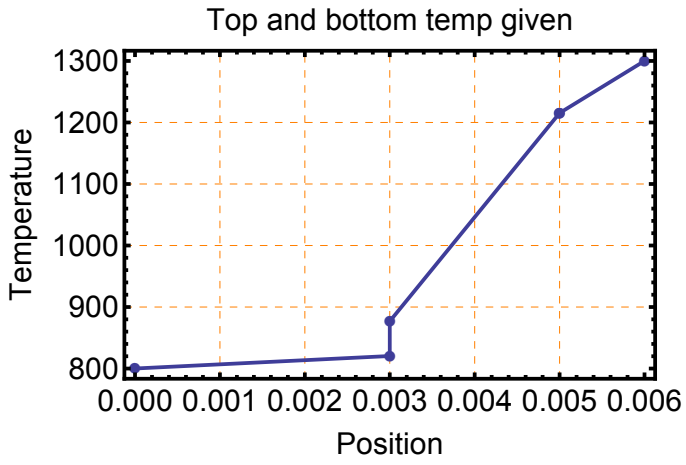

FIG. 2. Temperature distribution from the "two temperature boundary condition".

Note thay, if the temperature is known at the bottom of Layer2, we could get the same results by running the following boundary conditions:

$$\texttt{twoTbcs = } \{\{2,0,876.8\},\{3,1,1300.\}\}\texttt{; (*layer number, 0-bot,1-top, temp*)} \tag{30}$$

$$\texttt{ifc = } \{10^4, 10^7\} \tag{31}$$

That is, one can prescribe two temperatures at any locations within the stack, including those on *either* side of an interface with finite conductance.

Conversely, if we knew the top surface temperature was 1300^o and the heat flux was from top to bottom with magnitude $Q = 169,294$, we could run the "one temp and heat flux bc" analysis, as in:

```
Tout2 = TempSolQ[Multilayerex, ifc, {{3,1,1300},-169294}]
```
(32)

$$\{\{0.,800.001\},\{0.003,820.317\},\{0.003,876.748\},\{0.005,1215.34\},\{0.005,1215.35\},\{0.006,1300.\}\} \tag{33}$$

This result is identical to that obtained for the other boundary condition, as expected.

NOTE: Now that a temperature analysis has been conducted, the layer definitions are redefined with the results of the temperature analysis. Typing the following returns the layer definitions in matrix form, with each row representing the properties in one of the layers.

```
MatrixForm[Multilayer]
```
(34)

$$\begin{pmatrix} 0.003 & 2.\times 10^{11} & 0.3 & 0.000015 & \{70,70\} & 0 & 25. & \{800.,820.315\} \\ 0.002 & 4.\times 10^{10} & 0.2 & 0.000011 & \{70,70\} & 0 & 1. & \{876.747,1215.34\} \\ 0.001 & 1.\times 10^{11} & 0.2 & 0.000011 & \{70,70\} & 0 & 2. & \{1215.35,1300.\} \end{pmatrix} \tag{35}$$

That is, the current temperatures at the top and bottom of each layer have been re-defined to match the results of the thermal analysis.

D. Stress Results

To get stresses in the layer, we first run the analysis that gives us the total elongation and curvature of the stack. Examples are:

```
{e1p, k1p} = PlaneDeformation[Multilayer, 0, 0, 0]
```
(36)

```
{e1b, k1b} = BiaxialDeformation[Multilayer]
```
(37)

where the first command gets the elongation and curvature for the plane-strain case, while the second command gets the elongation and curvature for the biaxial case. These results are then used to compute the associated stresses, in this case in the $x-$direction (scaled by MPa):

```
Pstresses = GetStress[{e1p,k1p},{0,0}, Multilayer, 1, 10^6]
```
(38)

```
Bstresses = GetStress[{e1b,k1b},{e1b,k1b}, Multilayer, 1, 10^6]
```
(39)

$$\{\{\{0,-28.7655\},\{0.003,-18.7446\}\},\{\{0.003,162.367\},\{0.005,-11.5865\}\},\{\{0.005,-28.9895\},\{0.006,-130.041\}\}\}$$
$$\{\{\{0,-99.3313\},\{0.003,84.399\}\},\{\{0.003,133.796\},\{0.005,-20.8355\}\},\{\{0.005,-52.112\},\{0.006,-129.011\}\}\}$$

where the first data set are the stresses assuming plane-strain deformation, and the second data set are those assuming biaxial deformation. `ListPlot` is then used to plot the stresses in each layer, as shown below.

E. Energy Release Rate Results

The calculation of energy release rates are simply function calls to the ERR commands outlined above. **NOTE: If a thermal analysis has been run, the calculation will correspond to the ERR at the steady-state temperature distribution. If you want the ERR at a different temperature distribution, you have to simply directly specify the top/bottom temperatures for each layer, by re-writing the layer definitions.** The following command makes a list of ERRs at all the interfaces:

```
Gc = Table[{i, ClassicERR[Multilayer, i]}, {i,1,Length[Multilayer] - 1}]
```

Here is a `ListPlot` of the ERRs corresponding to three different deformation assumptions:

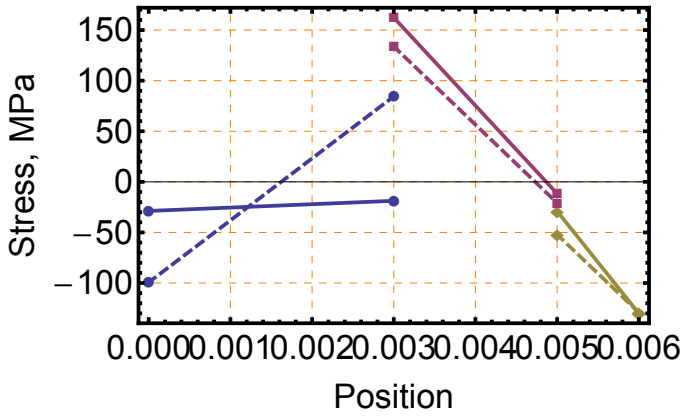

FIG. 3. Stress distribution at elevated temperatures from steady-state thermal analysis, in the $x-$direction. The dashed lines are the stresses assuming biaxial curvature is the deformation state, while the solid lines are the results from a plane-strain analysis.

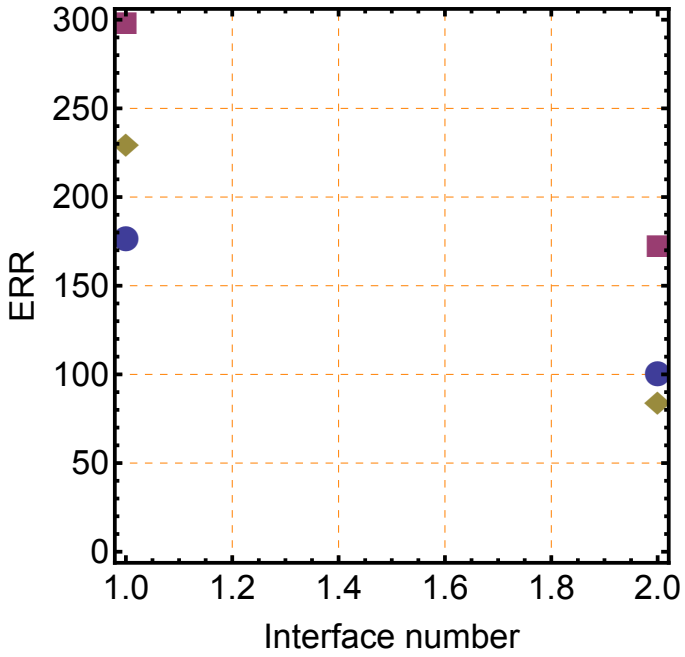

FIG. 4. ERRs at the two interfaces for the example problem, according to different deformation assumptions: (blue) - "ClassicERR" where the stress-state is biaxial ahead of the crack and only the $x-$direction stresses are relieved by cracking, (purple) - "BiaxialERR" where the deformation state is biaxial both ahead of the crack and behind the crack, (gold) - "PlaneStrainERR" where the out-of-plane strain is zero ahead of the crack and behind the crack.